Iiro Vanninen

Startup-kulttuurin vaikutus ohjelmistokehitysprosesseihin

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tietotekniikan koulutusohjelma

Insinöörityö

21.11.2014

Metropolia

Tekijä Otsikko Sivumäärä Aika	Iiro Vanninen Startup-kulttuurin vaikutus ohjelmistokehitysprosesseihin 26 sivua 21.11.2014
Tutkinto	Insinööri (AMK)
Koulutusohjelma	Tietotekniikan koulutusohjelma
Suuntautumisvaihtoehto	Ohjelmistotekniikka
Ohjaaja	Lehtori Olli Hämäläinen

Työn tavoitteena oli tarkastella startup-yritystä sekä startup-kulttuurin asettamia erityisvaatimuksia ohjelmistokehityksessä. Työssä esiteltiin startup-yritysmalli, jossa prosessit eroavat paikoitellen perinteisen yrityksen vastaavista huomattavasti. Tarkasteltiin tosielämän yrityksessä toteutettua casea, jossa ratkaistiin aloittelevan it-startupin teknologiaongelma. Tämän jälkeen analysoitiin startup-kulttuurin vaikutusta kyseisessä casessa.

Työssä kerrottiin yleisesti startup-yrittämisestä. Startup-yritykset yrittävät kasvaa markkinoilla erittäin nopeasti hyödyntämällä erikoisvahvuuksia. Kasvuyrityksen toimintaa leimaa innostus ja innovatiivisuus, mutta myös epävarmuus ja riskialttius. Rahat sekä aika ovat tiukassa, ja suurin osa epäonnistuu. Työssä käsiteltiin, millaiset teknologiaratkaisut ovat tällaisen yrityksen kannalta parhaita. Tutkittiin myös teknologia-alalla vallitsevia työskentelytapoja ja pohdittiin miten ne soveltuvat kasvuyrityksen tarpeisiin.

Insinöörityöhön kuului case it-alan startup-yrityksestä, CaterIt:istä. Yritys tarvitsi uudistuksia verkkopalvelunsa tiedonhallinnan työkaluihin. Koska uudet toiminnot piti saada mahdollisimman nopeasti käyttöön, ongelmaa lähdettiin ratkaisemaan hyvin tulosvetoisesti. Lopputuloksena yrityksen ongelmat saatiin ratkaistua ongelmien suuruuteen nähden nopeasti.

Tulosvetoisesta ohjelmistokehityksestä voi olla paljon hyötyä alati kasvavalle ja muuttuvalle yritykselle. Tavalliset tuotantoprosessit voivat olla liian kahlitsevia ja hitaita ympäristössä, jossa pitää tuottaa tauotta lisäarvoa. Varsinkin lyhyellä aikavälillä tuotettavat projektit hyötyvät tällaisesta lähestymistavasta. Tästä huolimatta minkään yrityksen ei tulisi laiminlyödä alalla hyviksi todettuja työskentelytapoja ja käytäntöjä pitkään. Ohjelmistoja tuotettaessa virheet korostuvat sekä pahenevat ajan myötä, ja varsinkin asiakkaalle tuotettaessa ne paisuvat moninkertaisiksi.

Avainsanat	PHP, HTML, startup, ohjelmistosuunnittelu, web-kehitys

Author Title Number of Pages Date	Iiro Vanninen Impact of a startup-culture in software development 26 pages 21 November 2014
Degree	Bachelor of Engineering
Degree Programme	Information Technology
Specialisation option	Software Engineering
Instructor	Olli Hämäläinen, Senior Lecturer

The purpose of this thesis was to inspect startup-companies and the special requirements created by a startup-culture in the field of software engineering. The thesis presented the startup-company model, in which processes can vary greatly from a traditional company. A real-world case was inspected. In the case, a young startup had a technology problem which was solved. The thesis also contained an analysis of the effects of startup-culture in this particular case.

This thesis had an introduction to startup-entrepreneurship. Startup-companies try to grow as fast as they can utilizing special attributes. The operations of a growth company are marked by enthusiasm and innovation, but also by uncertainty and risk. Time and money are scarce, and most companies like this will fail. The thesis discussed which technology solutions are the best for this type of company. There was a section about current working styles in the it-field. The thesis considered, which of these styles would be the most suitable for a growth company.

A case was included about a technology startup, CaterIt. The case company needed upgrades to the tools of its web service. The company was in a hurry to get the new features working, which is why the project was started from a very results-oriented standpoint. The problem was solved very quickly considering its size and complexity.

Results-oriented software development can be very useful for a growing and changing company. Normal production processes can be too restrictive and slow in an environment where more value has to be produced constantly. This is especially true for projects that are produced during a small time frame. However, no company should ignore the time-tested, good working processes and principles for very long. Mistakes tend to accumulate when software is developed without proper care, especially when developing for customers.

Keywords	PHP, HTML, startup, software engineering, web development

Sisällys

Lyhenteet

CSS	*Cascading Style Sheets.* Tyylitiedosto, jolla määritellään HTML-dokumentin ulkoasu.
FTP	*File Transfer Protocol.* TCP-protokollaa käyttävä tiedonsiirtomenetelmä.
HTML	*Hypertext Markup Language.* Avoimesti standardoitu kuvauskieli hypertekstin kuvaamiseen.
PDF	*Portable Document Format.* Dokumenttien järjestelmästä riippumattomaan esittämiseen kehitetty tiedostoformaatti.
PHP	*PHP: Hypertext Preprocessor.* Palvelinpuolen web-ohjelmointikieli.
SCP	*Secure Copy Protocol.* Protokolla kahden etätietokoneen väliseen, turvalliseen tiedostonsiirtoon.
SFTP	*Secure File Transfer Protocol.* Turvattu FTP-yhteys.
SQL	*Structured Query Language.* Standardoitu kyselykieli. Usein käytössä relaatiotietokannoissa.
URL	*Uniform Resource Locator.* Merkkijono, jolla osoitetaan jonkin tiedon paikka. Käytetään usein osoittamaan WWW-sivuja.
WinSCP	*Windows Secure Copy.* Avoimen lähdekoodin SFTP, SCP ja FTP pääteohjelma.

1 Johdanto

Startup-yrittämisen määrä on kasvanut huimasti viimeisen kymmenen vuoden aikana. Kirjoitushetkelläkin käynnissä oleva startup-buumi on nostanut kyseisen yritysmallin erittäin näkyväksi. Suomen yrityshistorian suurimpia menestystarinoita, kuten Supercell ja Rovio, on noussut viime vuosina startup-perusteista maailmanlaajuisiksi yrityksiksi. Vaikka samanlainen onnistuminen on useimmille startup-yrittäjille vain haave, kasvuyrittäjyys on helpompaa ja tuetumpaa kuin koskaan aikaisemmin.

Startup-yrittäminen asettaa erityisvaatimuksia yritystoiminnan jokaisella osa-alueella, myös teknologiapuolella. Tekniikan pitäisi olla nopeaa ja toimia luotettavasti, mutta samalla olla valmiina kasvuun. Lisäksi, koska usein ei olla vielä varmoja, mihin suuntaan yritys tulee kehittymään, teknologian vaihtuminen on tavallista.

Ohjelmistotuotannossa pitäisi aina pyrkiä tukevalla pohjalla lepäävään, hyvin dokumentoituun lopputulokseen. Startupeissa kuitenkin epävarmuus ja resurssien puute pakottavat improvisoimaan sekä tuottamaan nopeita tuloksia. ”Quick-and-dirty”-ratkaisut voivat osoittautua käytännöllisiksi lyhyellä aikavälillä. Myös työskentelytavat poikkeavat suurempien yritysten standardeista. Tämän insinöörityön tavoitteena on esitellä tekniikkaa ja ohjelmistokehitystä startup-ympäristössä.

Jotta lukija saadaan orientoiduttua aiheeseen, insinöörityöhön kuuluu esittely startup-yrityksistä ja ajatusmaailmasta startup-yrittämisen takana. Työhön kuuluu tosielämän case aloittelevan IT-alan startupin teknologiaongelman ratkaisusta. Esitellään casen yritys, sen ongelma ja ongelman ratkaisu. Tästä siirrytään yleisempään katsaukseen ohjelmistotuotannosta startupeissa. Tarkastellaan, millaisia vaatimuksia nopea kasvu ja epävarmuus yrityksen suunnasta aiheuttivat teknisellä puolella käytännössä.

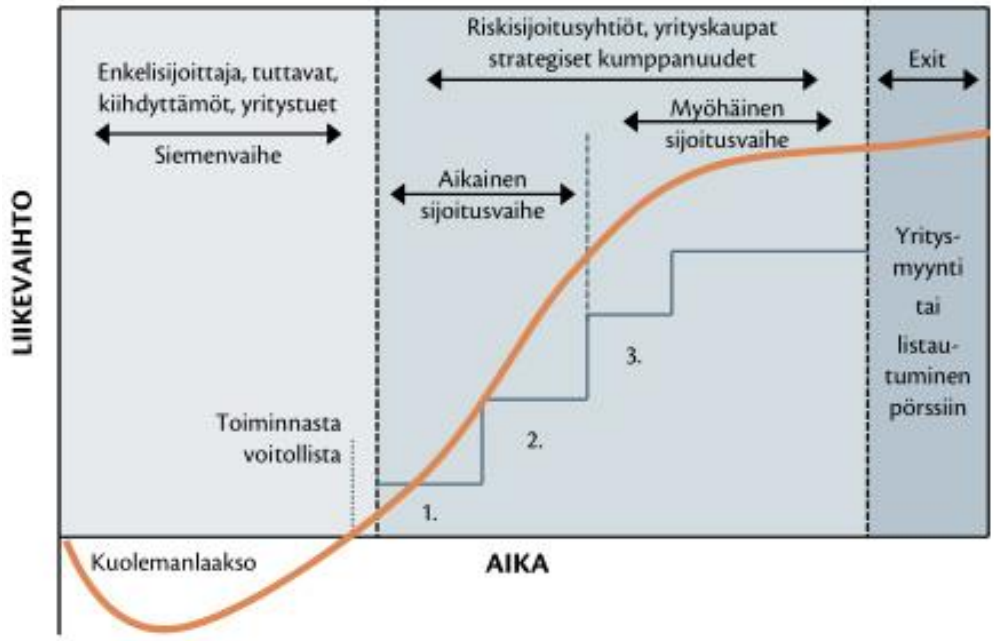
2 Startup-yritys

Startupit ovat nuoria, iältään yleensä alle viisivuotiaita yrityksiä, jotka tähtäävät räjähdysmäiseen kasvuun hyödyntämällä jotain uniikkia vahvuutta tai ominaisuutta. Tämä vahvuus voi olla jokin idea, tuote tai vaikkapa uusi tapa tehdä työtä. Startup-yrityksiin viitataankin usein nimikkeellä kasvuyritys. Neil Blumethal, Warby Parkerin toimitusjohtaja, luonnehti startup-yritystä sanoilla: ”Startup on yritys, joka tekee töitä ratkaistakseen ongelman, jonka ratkaisu ei ole selvä ja onnistuminen on epävarmaa.” [1.]

Startup-yrittämisessä keskitytään innostukseen, oivalluksiin ja osaamiseen paljon keskeisemmin kuin perinteisessä yrittämisessä. Tavoitteena on usein luoda ei vain uutta, vaan radikaalisti uutta. Startup-toiminta on kunnianhimoista, vapaata ja kansainvälistä, mutta riskialtista. Yhdysvalloissa tehdyn tutkimuksen mukaan vain 36 % teknologia-startupeista oli pystyssä neljän vuoden jälkeen liiketoiminnan aloittamisesta. [2.]

Startup-yrittäjät tekevät usein aluksi töitä kokonaan ilman palkkaa. Myöhemmin rahoitus saadaan riskirahoitusmallilla, joka tarkoittaa, etteivät yrityksen perustajat välttämättä sitoudu itse rahallisesti yhtiöönsä. Sijoittajina toimivat bisnesenkelit, yritykset ja joskus valtio. Tällä rahoitusmallilla yritetään jättää perustajille tilaa ottaa riskejä ja etsiä rohkeasti parasta toimintamallia yritykselle. Kuva 1 havainnollistaa tätä rahoitusprosessia. [3.]

Tuottoa tavoittelevan startupin tavoitteena on myynti isommalle yritykselle tai listaus pörssiin. Tämä tapahtuma ”exit” tapahtuu usein jo muutaman vuoden kuluessa, jos se on tapahtuakseen. Exitissä kaikki perustajat ja yhtiöön sijoittaneet tahot rikastuvat yleensä huomattavasti. Vuonna 2013 keskimääräisen startup-exitin arvo Yhdysvalloissa oli 242,9 miljoonaa dollaria. Keskimääräinen sijoitusten määrä kasvuyrityksiin oli tuolloin 41 miljoonaa. [4.]



Kuva 1. Startup-yrityksen rahoituksen aikajana [3.]

Ehkä yhdistävin tekijä minkä tahansa alan startupien välillä on niiden jatkuvasti muuttuva ympäristö ja toimintamalli. Startupin toimintamallin voidaankin nähdä sisältävän ainoastaan joukon oletuksia tuotteesta, toimintaympäristöstä ja markkinoista. Startup ei yleensä tiedä, mikä on sen paras tapa tehdä tulosta. Onkin tavallista, että startupilla ei vielä ole kannattavaa liiketoimintaa. Jatkuva muutos aiheuttaa paineita yrityksen jokaisella osa-alueella.

3 Katsaus ohjelmistokehitykseen startupissa

Ohjelmistokehitys startup-yrityksessä voi erota huomattavasti perinteisen yrityksen prosessista. Kaikki teknologiavalinnoista siihen, millaista itse koodi on, voi olla erilaista. Tässä luvussa perehdytään startup-kulttuurin aiheuttamiin erityisvaatimuksiin ohjelmistokehityksessä sekä perustellaan kasvuyrityksen näkökulmasta ohjelmistokehitykseen liittyviä valintoja.

3.1 Työskentelytapa

Startupeissa kaikki työntekijät työskentelevät usein rinnakkain harjoittelijoista perustajajäseniin. Ison firman työnteosta poiketen startupissa ei pidä navigoida läpi byrokratian saadakseen muutosta aikaan ja ideoitaan esille. Startup-maailmassa muutoksen keskellä moiseen ei yksinkertaisesti ole aikaa. [5.]

Startupeissa työskennellään hyvin tulosvetoisesti ja keskitytään siihen, mikä tuottaa mahdollisimman paljon lisäarvoa sillä hetkellä. Usein suuremmissa yrityksissä käsitellään asioita pitkällä tähtäimellä. Ajattelu keskittyy kestävien asioiden tuottamiseen esimerkiksi viiden vuoden aikavälillä. Startup hakee konkreettisia tuloksia jatkuvasti. Vertaukseksi voidaan ottaa seuraava: suuressa organisaatiossa ihmiset pitävät auton liikkeellä varmistamalla, että öljyt on vaihdettu ja renkaat ehjät. Startupissa täytyy käydä joskus ulkona työntämässä, pääasia että auto liikkuu eteenpäin. [6.]

Tällainen yrityksen toimintaympäristö asettaa haasteita ohjelmistokehityksessä. Perinteinen vesiputousmalli ei ole omiaan vastaamaan äkillisiin muutoksiin. Ikääntyneiden ohjelmistokehitysprosessien vastineeksi esiteltiin 2000-luvun alussa niin sanotut ketterät menetelmät (Agile methods). Agile-menetelmien käyttö on yleistynyt viime vuosien aikana valtavasti IT-yritysmaailmassa. Menetelmiä ei kehitetty varsinaisesti startupeille, mutta ne sopivat startuppien toimintaympäristöön erityisen hyvin.

Scrum

Scrum on yksi yleisimmin käytetyistä ketterän ohjelmistokehityksen viitekehyksistä. Scrum soveltuu hyvin ohjelmistokehitykseen, mutta on oikeastaan yleissopiva malli mille tahansa projektille. Virallinen Scrum Guide määritteleekin Scrumin seuraavasti: ”Scrum on monimutkaisten tuotteiden kehittämiseen ja ylläpitoon tarkoitettu viitekehys.” [7.] Kuten useimmissa Agile-menetelmissä, Scrumissa työskennellään toistavasti ja lisäävästi (iterative-incremental). Tämä tarkoittaa sitä, että tavoitteena oleva tuote kehittyy pikkuhiljaa täydellisemmäksi ja valmiimmaksi useiden, yleensä lyhyehköjen kehitysjaksojen aikana. Scrum perustuu empiiriseen projektinhallintateoriaan. Teoria käsittää sisällään kolme tukipilaria: läpinäkyvyys, tarkastelu ja sopeuttaminen. [7.]

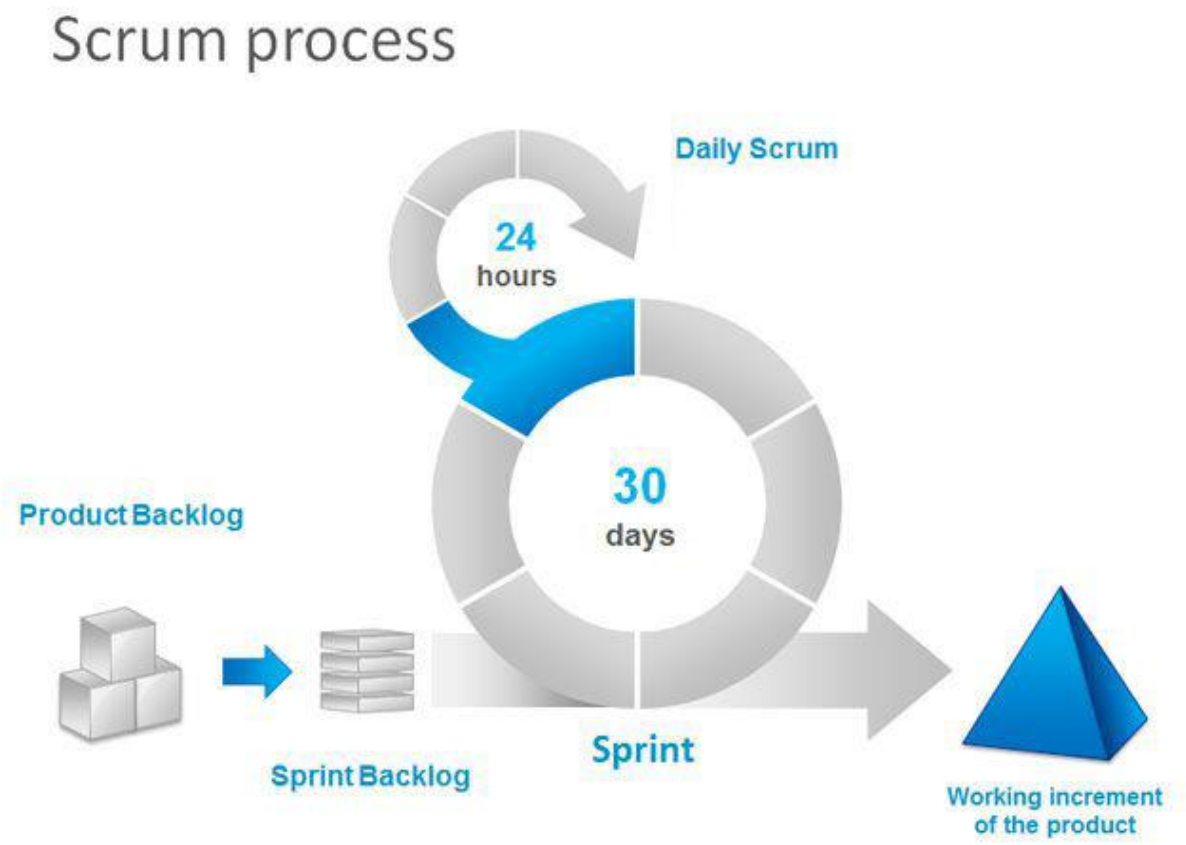
Läpinäkyvyys tarkoittaa prosessin ja työn näkymistä projektiryhmän sisällä ja sen ulkopuolella. Kaikilla osanottajilla pitäisi myös olla sama käsitys siitä, mitä työ pitää sisällään ja mitä sen valmistuminen tarkoittaa. *Tarkastelulla* haetaan jatkuvaa Scrum-projektin sekä sen etenemisen tarkastelua. Tällä yritetään löytää projektin epäkohdat mahdollisimman nopeasti. Tarkastelua pitäisi olla paljon, mutta ei niin paljon että se häiritsee itse työskentelyä. *Sopeuttaminen* tarkoittaa jatkuvaa prosessin sovittamista ennalta määrättyjen raja-arvojen sisälle ja prosessin turhien osien karsimista. Jos prosessin osia joudutaan muuttamaan syystä tai toisesta, prosessia ja resursseja tulee säätää niin kauan kunnes lopputulos näyttää taas suotavalta. [7.]

Nämä ovat periaatteet, jonka pohjalta Scrum toimii. Itse Scrum-viitekehys kostuu Scrum-sääntöjen mukaisesta Scrumtiimistä, Scrum-tapahtumista ja Scrumin tuotoksista. Scrumtiimi koostuu tuoteomistajasta, kehitystiimistä ja scrummasterista. Scrumtiimit on suunniteltu itseohjautuviksi ja sisältämään laajan joukon eri taitoja. Optimitilanteessa scrumtiimi on täysin riippumaton työryhmän ulkopuolisesta osaamisesta. Ideana on maksimoida joustavuus, luovuus ja tuottavuus. [7.]

Scrum-tapahtumia käytetään luomaan säännöllisyyttä ja minimoimaan palaverien tarve. Tapahtumiin kuuluvat sprintti, päiväpalaveri, sprintin katselmointi ja sprintin retrospektiivi. Sprintti on näistä tärkein, sillä se on Scrumin ydinprosessi ja se sisältää kaikki muut tapahtumat. Sprintti on enintään kuukauden pituinen aikaväli. Sen aikana tuotetaan käyttökelpoinen ja mahdollisesti julkaisukelpoinen tuoteversio. Jokainen sprintti on samanpituinen koko tuotekehityksen ajan. Sprintit ovat kuin pieniä projekteja projektin sisällä. Sprintit pidetään tarkoituksella lyhyinä, jotta tavoitteet pysyisivät

yksinkertaisina ja tuotteen määritelmä ei ehtisi muuttua. Kuvassa 2 näkyy sprintin lisäksi koko Scrum-prosessin kulku. [7.]

Scrumin tuotokset kuvaavat projektin työmäärää tai lisäarvoa. Niihin kuuluvat tuotteen kehitysjono, sprintin kehitysjono ja tuoteversio. Tuotokset ovat käytännössä dokumentteja, joilla pidetään kirjaa tavoitteista ja saavutuksista sekä tarkastellaan edistymistä. Scrumin tuotokset on rakennettu erityisesti läpinäkyvyyden maksimoimista ajatellen. Niiden perusteelta on helppoa tarkastella ja sopeuttaa tuotetta. [7.]



Kuva 2. Scrum-tuotantosykli [8.]

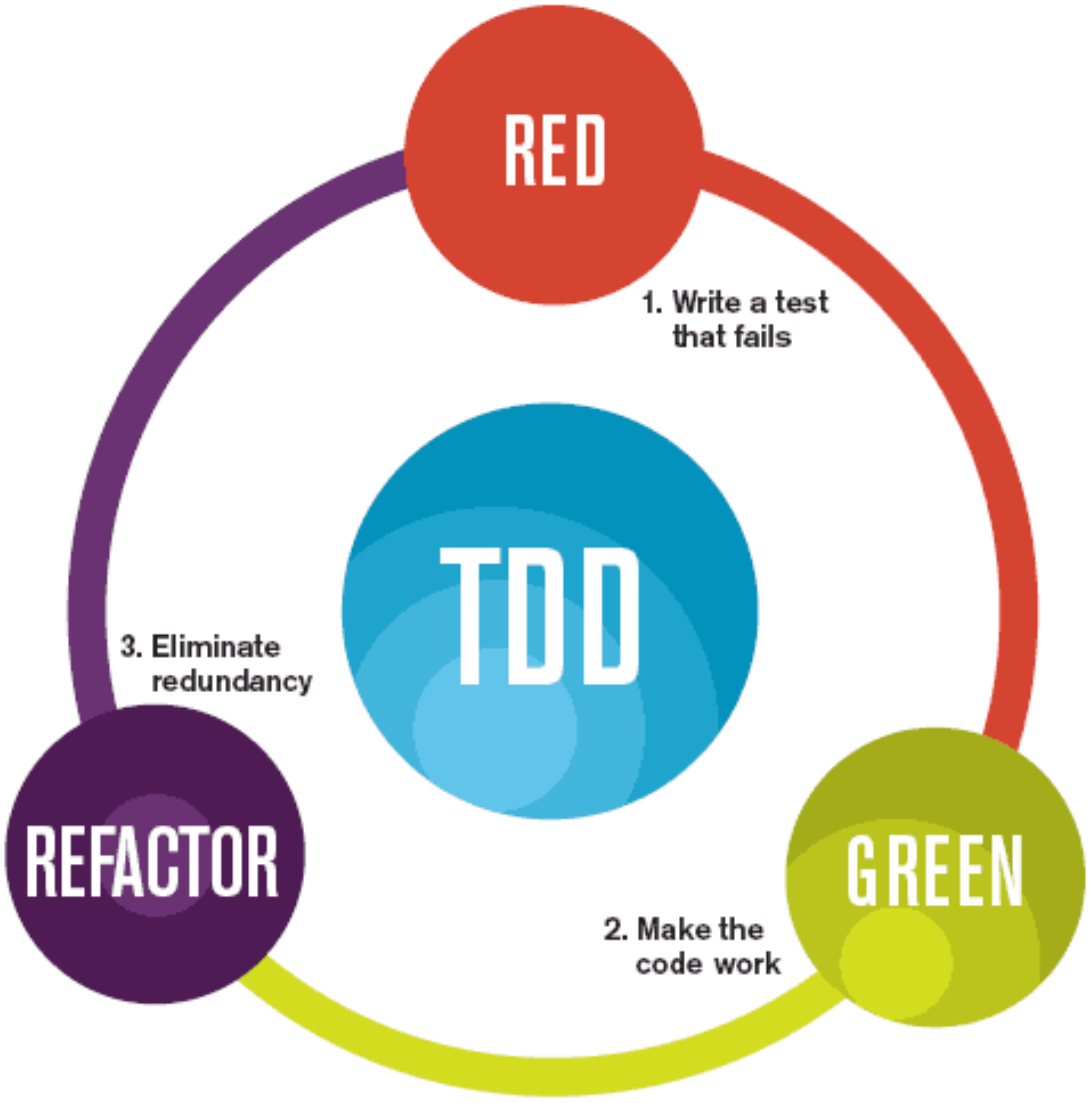
Lyhyet, inkrementaaliset kehitysjaksot soveltuvat ohjelmistokehitykseen erityisen hyvin. Koska vaatimukset ja tarpeet muuttuvat ohjelmistoprojektissa usein, eteneminen askel kerrallaan on järkevä lähestymistapa projektinhallintaan. Yksi Scrumin parhaista eduista on, että jatkuvasti tuotetaan jonkinlainen valmis tuote. Hyöty tästä on todella suuri ohjelmistokehityksessä, sillä se tarkoittaa, että sovelluksen eri osien kokonaisuuden täytyy olla integroitu sekä testattu sprintin lopussa. Integrointi ja testaus tulisi suorittaa usein missä tahansa ohjelmistoprojektissa, sillä näiden toimenpiteiden

lykkäys ennustaa katastrofia projektissa. Pahimmassa tapauksessa voi käydä niin, että projektin loppuvaiheessa lykättyä integrointia tehdessä sovelluksen osat eivät toimikkaan yhdessä. [9.]

Startupille Scrumin kaltaisesta viitekehyksestä voi olla paljon hyötyä. Tällainen projektinhallintamenetelmä voi tuoda startupin ohjelmistokehitykseen tarvittavaa kuria ja pitää kiireisen tiimin paremmin samalla sivulla. Toisaalta lisääntynyt paperityö ja kokoontumistarve saattavat hidastaa usein improvisoitua startup-prosessia. Startupille voitaisiinkin suositella sprinttiaikojen pitämistä lyhyinä ja ehkä jopa jonkinlaista karsittua versiota Scrumista. Esimerkiksi Scrumin tuotoksia voidaan supistaa vastaamaan startupin tarpeita.

Test Driven Development

TDD (Test Driven Development) on testivetoinen ohjelmistokehitysprosessi, joka keskittyy lyhyen kehityssyklin toistamiseen. Kuvan 3 mukaisessa syklissä testit kirjoitetaan ennen ohjelmakoodia, jonka jälkeen koodia kirjoitetaan kunnes testit menevät läpi. Ideana on edetä hyvin pieni askel kerrallaan jokaisessa syklissä. Vaikka askeleet ovat pieniä, kaikki tuotettu koodi on testattua ja valmista lähetettäväksi eteenpäin. TDD rohkaisee yksinkertaista suunnittelua ja vähentää virheiden määrää huomattavasti. [10.]



Kuva 3. TDD-kehitysprosessi vuokaaviona [11.]

TDD on mielenkiintoinen vaihtoehto startup-yritykselle. Yksikkötestien kirjoittamiseen saattaa kulua paljon kallisarvoista aikaa, eikä itse testeistä saada suoraa lisäarvoa. Startupin näkökulmasta tämä voi kuullostaa pahalta, mutta toisaalta yrityksen käteen jää varmasti toimiva tuote heti kun testit menevät läpi. Kasvuyrityksen sisäisissä projekteissa tämän edun arvo ei ole välttämättä vielä niin suuri, mutta ohjelmistoprojektia tuotettaessa asiakkaalle voidaan säästää paljon aikaa ja vaivaa. Tämä siksi, että minkä tahansa yrityksen sisäiseen käyttöön tuotetut ohjelmistot voidaan korjata suhteellisen nopeasti, vaikka virheitä olisi paljonkin. Asiakkaalle tuotettaessa viivästyksiä yleensä tulee enemmän, puhumattakaan asiakastyytyväisyyden ja tekijäyrityksen maineen kärsimisestä. Lisäksi TDD ei ole välttämättä niin hidas miltä se vaikuttaa. Sen oppimiskynnys on suuri, mutta kynnyksen

jälkeen prosessi nopeutuu huomattavasti. TDD on myös tunnettu ohjelmoijien taitoja kasvattavana työskentelytapana, asia joka jokaisen ohjelmistoja kehittävän yrityksen kannattaisi ottaa huomioon. Näiden seikkojen valossa ja tilanteesta riippuen, varsinkin jos työntekijät ovat kokeneita TDD:n käyttäjiä, voidaan tehdä johtopäätös, että TDD on varteenotettava työskentelytapa startup-ympäristössä. [11.]

3.2 Käytetyt ohjelmointikielet ja teknologiat

On tavallista, että startup-yrittäjät tekevät usein aluksi töitä kokonaan ilman palkkaa. Uuden IT-startupin teknologiavalintoja leimaa siksi yleensä niiden hinta tai pikemminkin sen puute. Aloittelevalla teknologiastartupilla on ehkä eniten syytä kaikista aloittelevista yrityksistä valita mahdollisimman kustannustehokas teknologia-alusta. Moni hyvä ja todistetusti toimiva teknologia on ilmainen, joten tämä ei yleensä ole ongelma. Toisaalta on monia syitä valita maksullisia teknologioita ilmaisten sijasta. Maksullisilla teknologioilla on esimerkiksi yleensä parempi tuki kuin ilmaisilla. Valintaan saattaa myös vaikuttaa ohjelmistokehityksen yleinen ilmapiiri: kun teknologia on suosiossa, sille on helpompi löytää kehittäjiä. Nämä syyt ovat kuitenkin parempia yrityksessä, jolla on pääomaa ja toimiva bisnesmalli. Startupissa raha on yleensä niin tiukassa, että ilmaisien ratkaisujen käyttöönottoa kannattaa harkita vakavasti. Teknologiaa valitessa kannattaa pitää mielessä, että maksullisilla sekä maksuttomilla vaihtoehdoilla on molemmilla hyvät ja huonot puolensa. Hyvän idean saa toteutettua melkein millä vain teknologialla. [12.]

Seuraavat teknologiat on valittu esiteltäväksi, koska niitä kaikkia käytettiin luvun 4 casessa. Ne myös edustavat todella kustannustehokasta linjaa, sillä ne ovat kaikki ilmaisia. Kaikilla näistä teknologioista on hyvä tuki, ja ne ovat laajasti käytössä.

HTML5

HTML (Hypertext Markup Language) on parhaiten web-sivujen perustana tunnettu, standardoitu hypertekstin merkintäkieli. HTML-dokumentti koostuu elementeistä, jotka määritellään tunnisteilla (tags). Web-selainten tehtävänä on tulkita HTML-kieltä ja esittää se käyttäjälle. HTML:ää käytetään siis kuvaamaan sivun tarkoitusta ja sisältöä. Tähän selaimen sekä HTML-kielen väliseen yhteistyöhön perustuu erittäin suuri osa koko WWW-teknologiasta. [13.]

Kirjoitushetkellä HTML5 on uusin HTML-versio. HTML5 lisäsi monia toimintoja ja suoraviivaisti entistä HTML-merkkausta aikaisemmasta versiosta. Uusina toimintoina tulivat esim. canvas-, video- ja audio-tunnisteet, jotka helpottavat multimedian toistoa entisestään. HTML5 on helppo ottaa käyttöön mistä tahansa aiemmasta HTML-versiosta. [14.]

CSS

CSS (Cascading Style Sheets) on WWW-dokumenteille kehitetty, vaikka muissakin rakenteellisissa dokumenteissa käytetty tyyliohjeiden laji. CSS-dokumentti koostuu tyyliehdotuksista, jotka koskevat jonkin tai joidenkin dokumenttien esitystapaa. Ehdotukset eivät ole ehdottomia, joten ne voidaan haluttaessa kiertää. Yleensä kuitenkin, ainakin WWW-kontekstissa, ne on määrittänyt sivuston ylläpitäjä tavalla, joka on mietitty käyttökokemuksen optimoimiseksi tai brändi-ilmeen säilyttämiseksi. CSS:n käyttö tuo mukanaan useita etuja, ja se on nykyään osa miltein jokaisen modernin web-sivun kokonaisuutta. [15.]

JavaScript

JavaScript on laajasti käytetty ohjelmointikieli jota käytetään useimmiten web-kehittämisen yhteydessä. Suosittu webkehittäjien informaatiosivusto W3Schools kutsuu sitä ”maailman suosituimmaksi ohjelmointikieleksi”. JavaScriptillä pystyy web-kontekstissa muuttamaan HTML-elementtejä, mikä on JavaScriptin yleisin käyttötapa. Usein JavaScriptiä käytetään parantamaan web-sivujen käytettävyyttä ja lisäämään toimintoja, joita ei pelkällä HTML-koodilla pystytä toteuttamaan. [16.]

PHP

PHP (PHP: Hypertext Preprocessor) on web-käyttöön tähdätty palvelinpuolen ohjelmointikieli. Se on tehokas työkalu dynaamisten ja interaktiivisten web-sivujen nopeaan luontiin. PHP:n kehitys aloitettiin henkilökohtaisena projektina 1994. Nykyään PHP on laajasti käytetty myös kaupallisissa ratkaisuissa. PHP-tiedosto voi sisältää tekstiä, HTML:ää, CSS:ää, JavaScriptia ja PHP-koodia. [17.]

PostgreSQL

PostgreSQL on vapaan lähdekoodin olio-relaatiotietokantapalvelin. Sitä kehittää suuri määrä yksittäisiä ohjelmoijia ympäri maailmaa, joita johtaa pieni ryhmä pääkehittäjiä. Pääkehittäjät valitsevat, mitkä ominaisuudet lopulta sisällytetään koodipuuhun. PostgreSQL on vaihtoehto muille lukuisille ilmaisille sekä kaupallisille tietokantaratkaisuille. Suosituin vertailu on ollut PostgreSQL:n ja MySQL:n välillä, sillä ne ovat käytetyimmät avoimen lähdekoodin tietokantaratkaisut. MySQL on saanut kehuja helppokäyttöisyydestään, kun taas PostgreSQL on kunnostautunut ominaisuuksien määrässä ja luotettavuudessa. [18; 19.]

3.3 Käytetyt työkalut

Startupissa käytetyistä työkaluista voidaan tehdä samanlaiset johtopäätökset kuin ohjelmointikielistä ja teknologioista. Ilmainen on aina hyvä, mutta joissain tapauksissa voi olla järkevämpää ostaa lisenssejä. Yksittäisten työkalujen lisenssit ovat usein halvempia kuin kokonaisten teknologioiden. Näitä työkaluja käytettiin luvun 4 casessa. Työkalut ovat ilmaisia.

Notepad++

Notepad++ on ohjelmoijille suunnattu mutta muuhunkin käyttöön soveltuva tekstieditori. Se on tällä hetkellä kehittäjien keskuudessa yksi suosituimmista vaihtoehdoista. Notepad++ tarjoaa useita raskaan tekstin muokkaukseen hyödyllisiä ominaisuuksia, kuten välilehdet, tekstin värikorostukset ja sessioiden tallentamisen. Editori on suunniteltu ohjelmistokehittäjiä varten, joten siinä on oletuksena päällekytkettynä useita ohjelmointia helpottavia ominaisuuksia.

WinSCP

WinSCP (Windows Secure Copy) on avoimen lähdekoodin SFTP, SCP ja FTP pääteohjelma. Se on suunniteltu tarjoamaan turvallisen yhteydenoton paikallisen (local) ja etätietokoneen (remote) välillä. Casessa työkalua käytettiin pääpalvelimelle yhdistämisessä ja tiedostojen päivityksessä.

3.4 Versionhallinta

Versionhallinnaksi kutsutaan mitä tahansa menetelmää tai ohjelmistoa, joka mahdollistaa töiden eri versioiden hallitsemisen. Ohjelmistokehityksen alalle on muodostunut standardiksi jonkinlaisen versionhallinnan käyttäminen melkein jokaisessa projektissa. Tämä on ymmärrettävää: versionhallinnan sisällyttäminen ohjelmistokehitysprosessiin tehostaa työntekoa huomattavasti sekä yksilötasolla että ryhmässä. Versionhallinta vähentää turhan työn määrää ja auttaa ohjelmoijaa oppimaan vanhoista virheistä. Useimmissa versionhallintaratkaisuissa on jonkinlainen yhteinen säilytyspaikka. Tämä helpottaa tiimin työskentelyä saman projektin parissa huomattavasti, kun muiden työtä voidaan kommentoida ja työhön voidaan tehdä muutoksia. Nämä edut saavutetaan pienellä vaivalla. Versionhallintaohjelmistoissa on yleensä pieni oppimiskynnys, mutta kynnyksen yli päästyään työntekijän työteho kasvaa huomattavasti. Oli yritys sitten perinteinen tai startup-mallinen, versionhallinta on erinomainen resurssi ohjelmistoprojekteihin. [20.]

Git

Git on yksi yleisimmin käytetyistä versionhallintajärjestelmistä. Se on avoimen lähdekoodin ohjelmisto, jonka toimintaperiaate perustuu paikalliseen ja ulkoiseen säilytyspaikkaan (repository). Usein Gitin käyttö yritysympäristössä toimii seuraavasti: paikallinen säilytyspaikka sisältää kaikki yksittäisen ohjelmoijan eri muutokset ja versiot. Ulkoisessa säilytyspaikassa sen sijaan sijaitsee projektin yhteinen hakemisto, joka on saatavilla kaikille kehittäjille. Gitille on tehty oma, kaikkien käytettävissä oleva ulkoinen säilytyspaikka, GitHub.

4 Case CaterIt: tiedonhallintatyökalujen päivitys

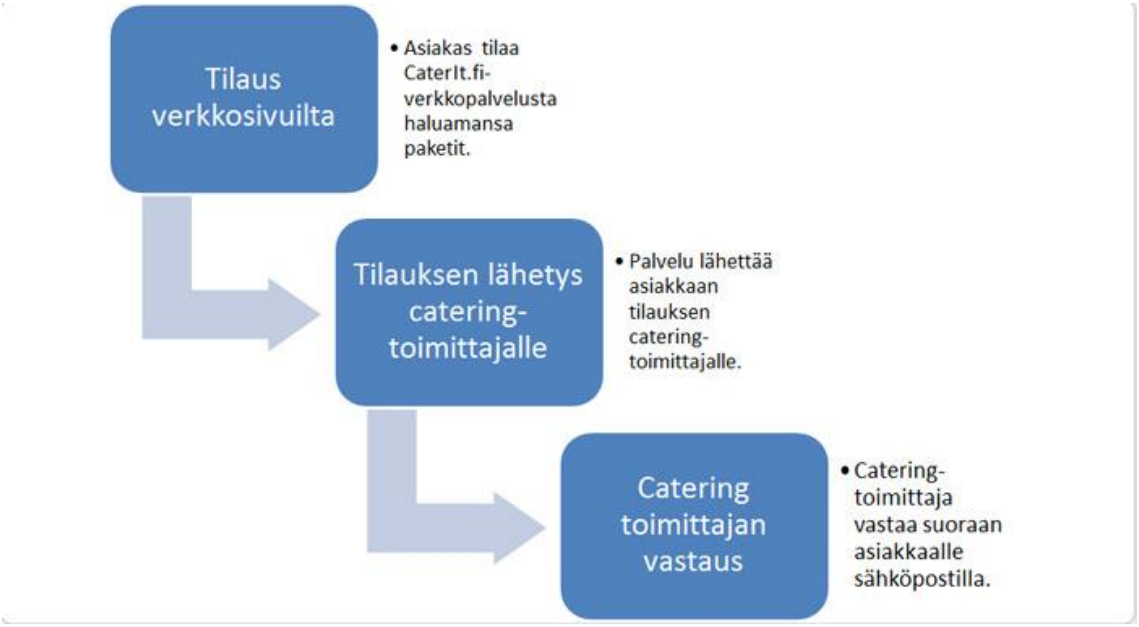
4.1 CaterIt-yritys ja -verkkopalvelu

Tämä case käsittelee työskentelyä CaterIt-nimisessä IT-alan startupissa kesällä 2014. Noin puoli vuotta vanhalla yrityksellä oli viisi työntekijää ja tilat Espoon Suomenojalla. Bisnesideana on catering-palvelujen välittäminen yrityksen verkkopalvelun kautta. Yrityksellä on jonkin verran liiketoimintaa, mutta ei vielä kannattavaksi asti.

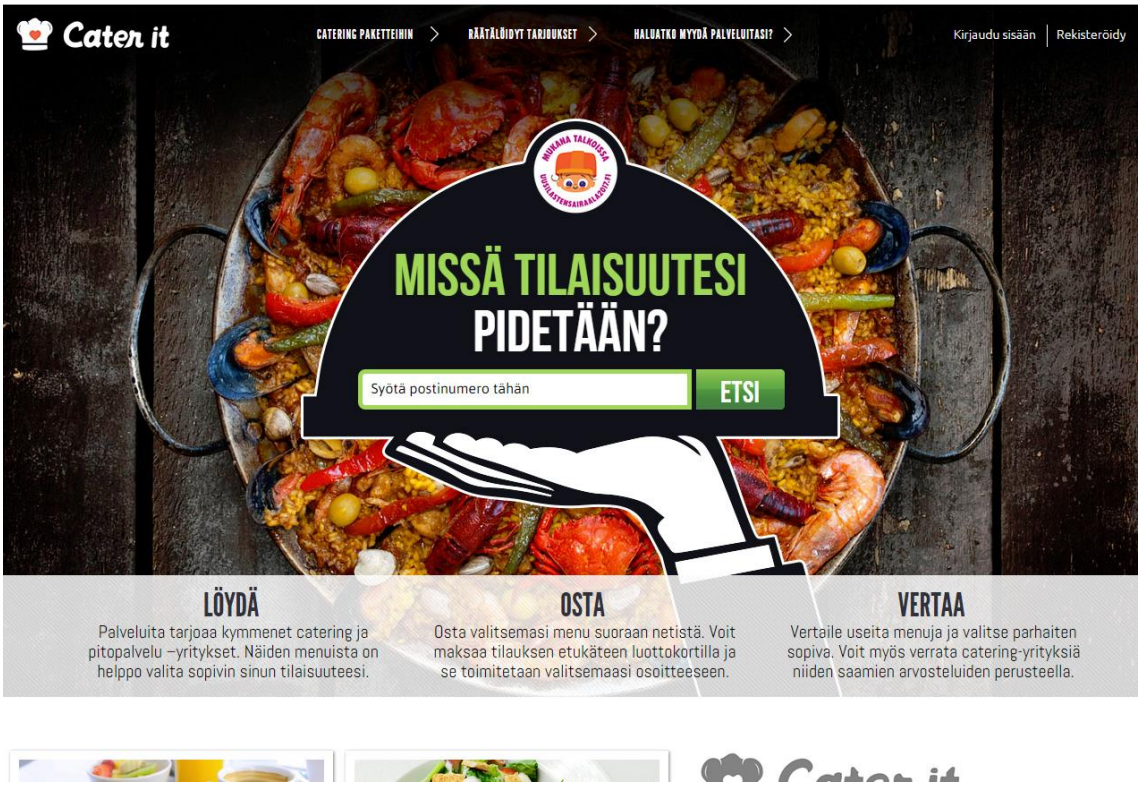
CaterIt toimii catering-palveluiden välittäjänä. Yritys ei siis itse järjestä palveluja, vaan yrittää tuoda asiakkailleen lisäarvoa tekemällä palvelujen tilaamisen ja tarjoamisen mahdollisimman helpoksi. Catering-palvelujen omat verkkosivut ja varsinkin tuotelistaukset ovat usein epäselviä eivätkä kovin moderneja. Kokoamalla tuotteet ja hinnat yhteen paikkaan sekä suoraviivaistamalla tilausprosessia CaterIt katsoo tuovansa markkinoille uudenlaisen, paremman tavan tilata cateringia. CaterIt ottaa 10 %:n välityspalkkion kaikista tilauksista. Yrityksen tavoitteena on muuttaa tilausprosessi täysin automaattiseksi.

Verkkopalvelu Caterit.fi koostuu kirjoitushetkellä kahdesta osa-alueesta: julkisesta puolesta ja käyttäjäpuolesta. Verkkopalvelun pääsivu on kuvan 5 mukainen ja osa julkista puolta. Tätä puolta pystyy käyttämään kuka vain käyttäjä catering-palveluiden tilaamiseksi eri toimittajilta. Verkkosivuille kirjoitetaan postinumero, johon palveluja halutaan toimitettavaksi. Tämän jälkeen sivut näyttävät, mitä palvelupaketteja kyseiselle alueelle voidaan tuottaa. Paketteihin pääsee myös käsiksi antamatta postinumeroa, mutta se tarkoittaa, että kaikki tarkasteltavat paketit eivät välttämättä ole tarjolla halutulle alueelle. Käyttäjä voi myös tehdä ns. räätälöidyn tilauksen pääsivulta.

Kun asiakas on tilannut haluamansa tuotteet verkkopalvelusta, palvelu lähettää automaattisesti tilauksen sähköpostilla eteenpäin toimittajalle. Toimittaja toimittaa asiakkaalle sovitun palvelun. Räätälöidyssä tilauksessa CaterIt:n henkilökunta hoitaa sähköpostiviestimisellä kanssakäynnin toimittajan kanssa alusta loppuun. Kuvasta 4 voi tarkastella tilausprosessia visuaalisessa muodossa.



Kuva 4. Caterit.fi:n-tilausprosessi.

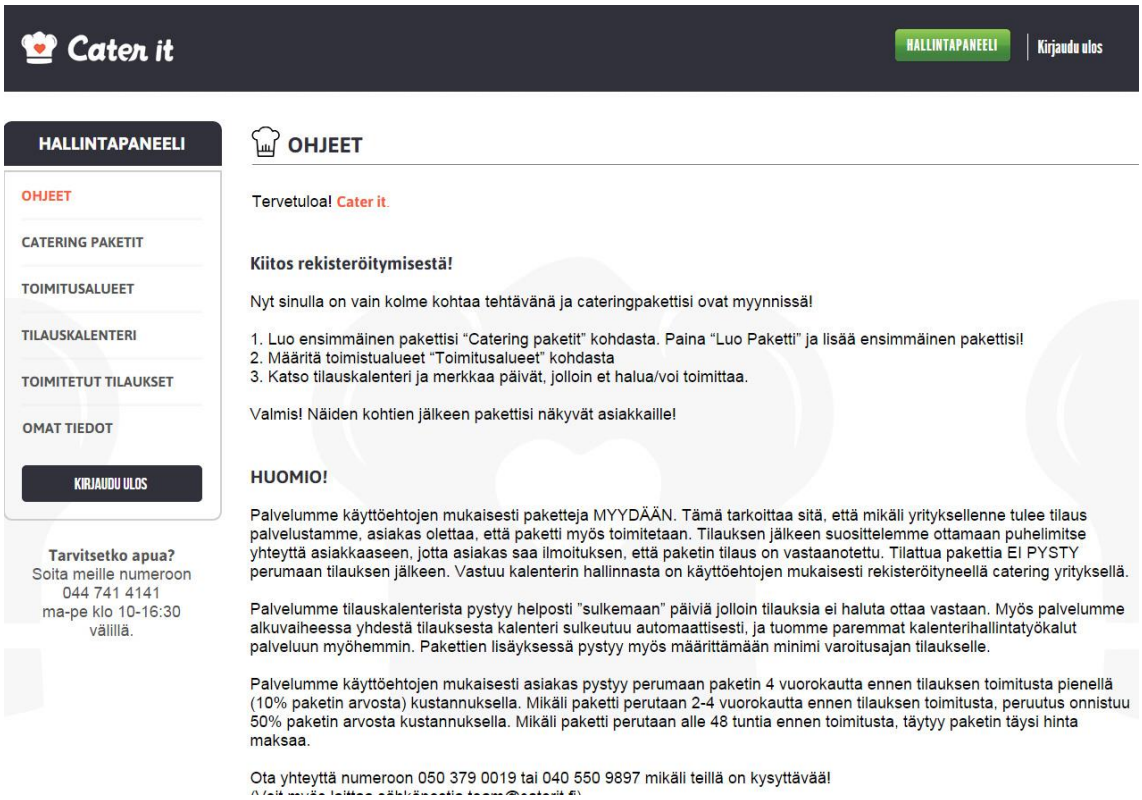


Kuva 5. Caterit.fi:n etusivu.

Käyttäjäpuoli on tarkoitettu catering-toimittajien käytettäväksi. Toimittajat pystyvät rekisteröitymään sivuille rekisteröintilomakkeen kautta, jonka jälkeen heille aukeaa pääsy käyttäjäpuolelle. Käyttäjäpuolella toimittajalle tärkein toiminto on mahdollisuus

lisätä sivuille omia catering-paketteja, mutta sieltä pystyy myös määrittelemään mm. toiminta-alueen ja näkemään toimitetut tilaukset. CaterItin henkilökunta käyttää käyttäjäpuolta omien, peruskäyttäjälle näkymättömien työkalujensa käyttöön sekä palvelun testaamiseen.

CaterItin työntekijöille oli rakennettu omia työkaluja helpottamaan tietojen hakemista ja tarkastelua. Kuten oli tarkoitus, näihin työkaluihin ei tavallinen verkkopalvelun käyttäjä päässyt käsiksi laisinkaan, vaan ne sijaitsivat ns. ”kulissien takana” käyttäjäpuolella. Näiden työkalujen käyttö oli mahdollistettu vain tietyille käyttäjille, käytännössä siis CaterItin työntekijöille.



Kuva 6. Caterit.fi:n käyttäjäpuolen näkymä.

4.2 Ongelma

Yrityksellä oli ongelmia tiedonhallinnan kanssa. Uutta tietoa tilauksista, asiakkaista, toimittajista ja itse verkkosivuista saatiin jatkuvasti ja tieto syötettiin automaattisesti yrityksen tietokantaan verkkopalvelun kautta. Yrityksen myyjät käyttivät näitä tietoja verkkopalvelun toiminnan tarkastelemiseen, laskuttamiseen ja yritysstrategian

suunnitteluun. Tiedon hakeminen kannasta oli kuitenkin kömpelöä: myyjien piti käyttää SQL-kieltä saadakseen tietoa kannasta tai tutkia vanhoja Excel-taulukoita, joihin tietoa oli myös säilötty. Saadakseen tietoa SQL-kielellä myyjien piti ensin kirjautua palvelimen ylläpitosivuille, jotka sijaitsivat ulkoisessa ylläpitopalvelussa. Tämän jälkeen piti ylläpitosivuilla avata phpMyAdmin, josta piti vielä löytää oikea kenttä SQL-kieltä varten. Excel-taulukoissa ongelmana oli niiden suuri määrä: taulukkotiedostoja oli satoja, eikä aina oltu varmoja, missä niistä tarvittavat tiedot sijaitsivat. Kaiken lisäksi taulukkotiedostoista oli eri versioita eri paikoissa. Kummatkin vaihtoehdot olivat siis hitaita, varsinkin SQL:n käyttö, jota jarrutti entisestään myyjien tietoteknisten taitojen puute. Koska myyjät käyttivät caterit.fi-verkkopalvelua jatkuvasti ja se oli heille tuttu, päädyttiin johtopäätökseen, että olisi parasta saada haettua tietoa jollain tavalla verkkopalvelun kautta.

Hitaan tiedonhaun ratkaisemiseksi oli jo ennen tämän projektin alkua kehitetty useita PHP-työkaluja, jotka hakivat tiedot kätevästi myyjien näkyviin ja joita pystyttiin käyttämään suoraan verkkosivujen kautta. Nämä yrityksen sisäiset työkalut olivat kuitenkin hajallaan yrityksen palvelimella. Työkaluihin oli hankala päästä käsiksi. Missään ei ollut minkäänlaista listausta työkaluista, ja useassa työkalussa tehtiin turhaan samoja asioita kuin toisissa.

Jos esimerkiksi myyjä halusi nähdä jonkun yksittäisen tilauksen tiedot, hänen piti ensin kirjautua sisään CaterIt-verkkopalveluun. Tämän jälkeen myyjän piti tietää oikean aputyökalun täsmällinen tiedostonimi, joka piti lisätä käyttäjäpuolen URL-osoitteen perään. Tämä tiedostonimi oli tiedoston nimi palvelimella, johon työkalu oli rakennettu. Myyjä kirjoitti uuden, pidennetyn osoitteen suoraan selaimen URL-kenttään (esim. ”http://caterit.fi/users/xxx_admin_orders_tool.php”) jonka jälkeen työkalu aukesi esiin. Nämä lisättävät merkkijonot piti muistaa ulkoa. Tällainen tapa päästä käsiksi työkaluihin oli erittäin kömpelö.

Yrityksen myyjillä oli myös esiintynyt tarve päästä vaikuttamaan itse suoraan tietokantaan. Aiemmin kaikki manuaaliset päivitykset piti hoitaa yrityksen it-puolen ihmisten kautta. Tämä ei ollut ongelma aikaisemmin, koska päivityksiä ei tullut kuin harvoin. Nyt kuitenkin tietokantaan oli tehty rakenteellisia muutoksia, ja muutosten myötä uusiin tietokannan alueisiin vaadittiin päivityksiä usein. Muutokset vaikuttivat jokaiseen tilaukseen, ja tilauksia oli liikaa, jotta jokaisen niiden kohdalla olisi voitu häiritä it-puolen työntekijöitä. Missään aikaisemmassa aputyökalussa ei ollut

tietokantaan vaikuttamisen mahdollisuutta, vaan ne oli tehty pelkästään tietokannan tarkasteluun.

Tarvittiin keskitetty luettelo työkaluista linkkien kera sekä uusi työkalu, joka yhdistäisi tärkeimmät tiedot yhden sivun alle sekä tarjoaisi mahdollisuuden vaikuttaa suoraan tietokantaan. Myyjillä ei ollut kuin perustason tietotekninen tietämys, joten tilanteeseen vaadittiin helppokäyttöistä ratkaisua.

Taulukko 1. Kehityssuunnitelma ja vaikutukset.

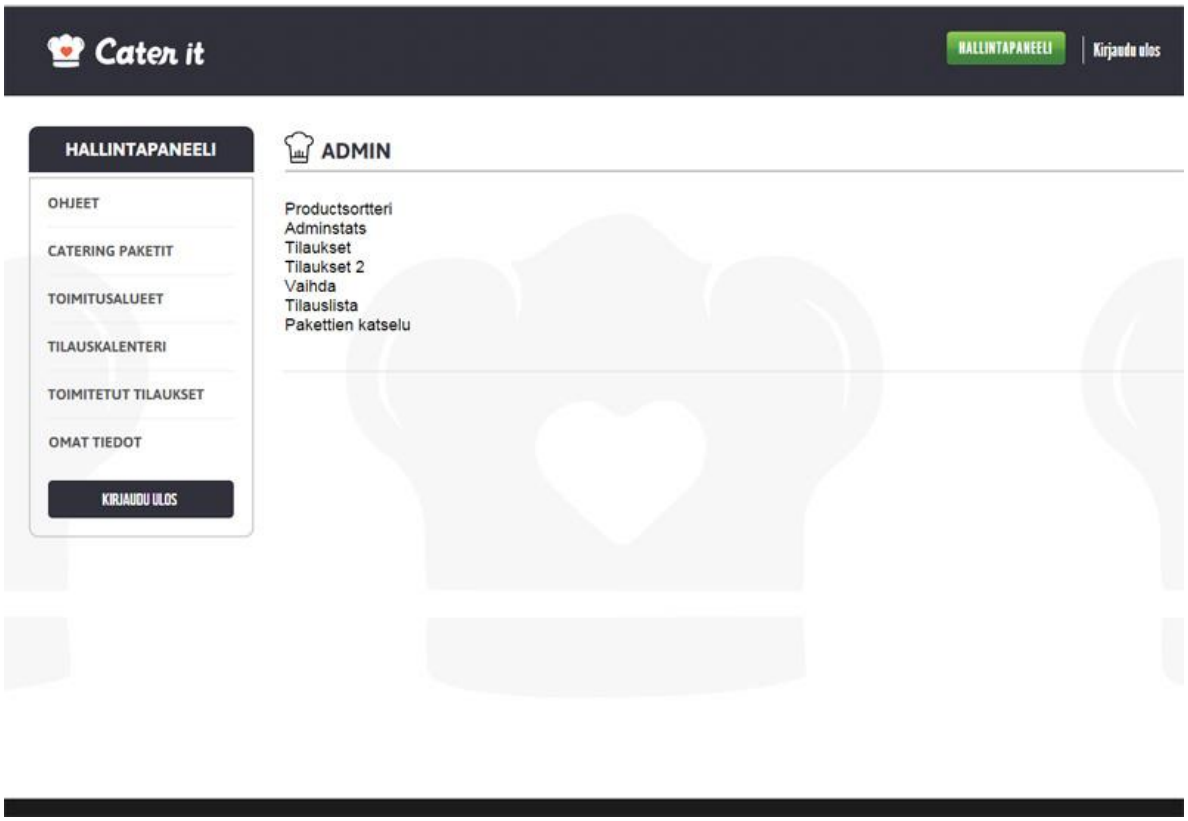
Ongelma	**Kehitysidea**	**Vaikutus**
Työkaluihin käsiksi pääseminen vaikeaa.	Koostesivun toteutus työkaluille.	Työkalujen käytön ja löytämisen helpottuminen.
Työkaluissa samoja ominaisuuksia tarpeettomasti.	Työkalujen yhdistäminen yhteen, kaikki tärkeimmät toiminnot kattavaan työkaluun.	Ajan säästyminen, tietokannan käsittelyn helpottuminen, myyjien työn yksinkertaistuminen.
Tietokantaan vaikuttaminen peruskäyttäjälle hidasta ja vaikeaa.	Tietokantaan vaikuttavan toiminnon lisäys johonkin työkaluun (tilausten käsittelyn tila).	Tietokannan nopean muokkauksen mahdollistuminen peruskäyttäjille (tilausten käsittelyn tilaa voi vaihtaa käsitelty/ei käsitelty).

4.3 Ratkaisu

Ongelmaa lähdettiin ratkaisemaan. Ensin tehtiin kunnollinen koostesivusto, johon kaikki työkalut kerättiin. Sivulle tuli linkit kaikkiin olemassaoleviin työkaluihin. Tälle sivulle, jota

kutsuttiin ”admin”-sivustoksi, pääsivät kaikki työntekijät sisäänkirjautumisen jälkeen linkin avulla. Sivu toteutettiin luonnollisesti PHP:lla. Kun sivu oli testattu ensin betapalvelimella, se lähetettiin caterit.fi livepalvelimelle. Integroinnissa caterit.fi-sivuston käyttäjäpuoleen ei ollut mitään ongelmia.

Jo tällaisen pienen koostesivuston tekeminen nopeutti työkalujen käyttöä huomattavasti. Huomattiin myös, että työkaluja oli tarjolla enemmän kuin muistettiinkaan: osa työkalujen nimistä oli unohtunut peruskäyttäjiltä, joten niihin ei päästy enää käsiksi. Nyt ne olivat taas saatavilla. Kuva 7 näyttää valmiin koostesivun.



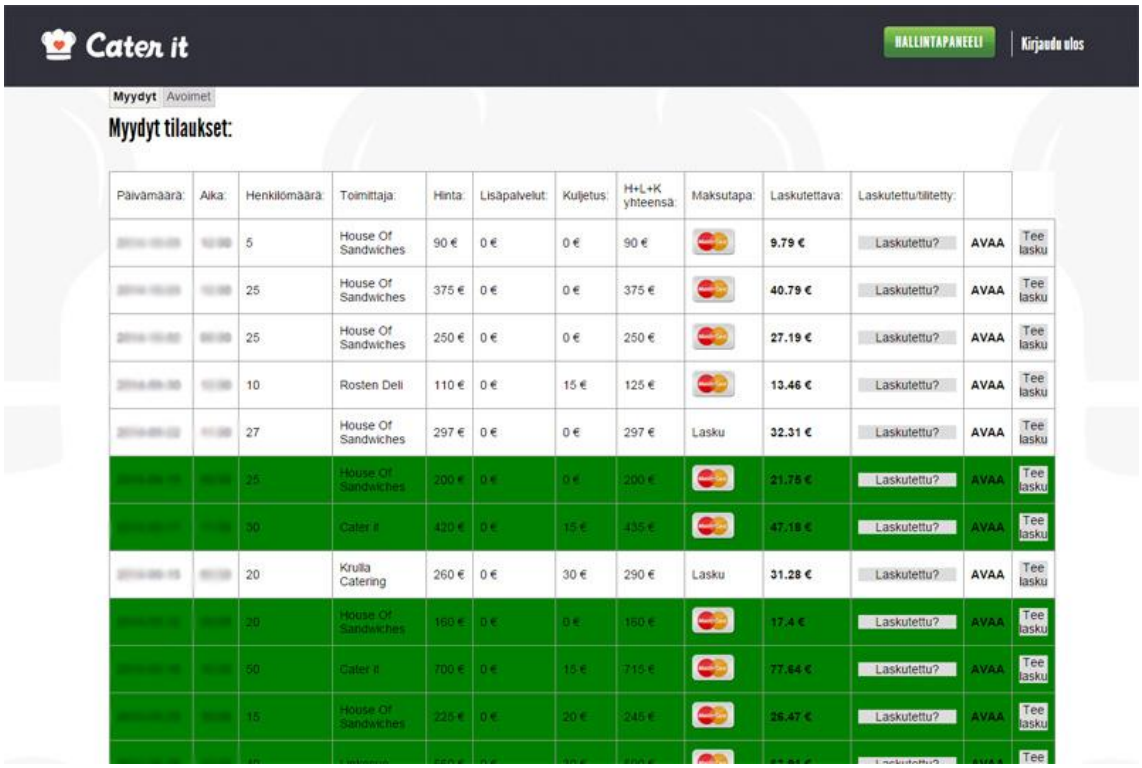
Kuva 7. Uusi Admin-portaali olemassaoleviin työkaluihin.

Aloitettiin yhdistetyn työkalun rakentaminen. Myyjät kertoivat, mitkä tiedot vanhoista työkaluista olivat tärkeimpiä ja asettivat toivomuksia uusista toiminnoista. Tärkeimmiksi tiedoiksi osoittautuivat asiakkaiden maksu- ja toimitustiedot. Uutena ominaisuutena toivottiin mahdollisuutta saada sarake käsitellyistä tilauksista sekä mahdollisuus muuttaa kyseistä tietoa suoraan työkalusta tietokantaan.

Työkalun tekninen toteutus voitiin pitää suhteellisen yksinkertaisena. Yrityksen web-sovellus oli rakennettu PHP-pohjalle. PHP:n piti siis olla lähtökohtana kehitykselle, joka soveltuikin hyvin tämänkaltaiseen projektiin. Työkalu toteutettiin PHP-, JavaScript- ja SQL-yhdistelmällä. Jako oli selvä: PHP- ja SQL-osuudet hoitivat palvelinpuolen toiminnot, ja JavaScript teki sivuista käytettävämmät selainpuolella.

Työkalun kehityksessä koitettiin pysytellä mahdollisimman yksinkertaisessa ulkoasussa ja maksimoida käyttönopeus sekä selkeys. Koska kyse oli pienen yrityksen sisäisestä työkalusta, ei kehenkään tarvinnut tehdä vaikutusta hienolla grafiikalla vaan pystyttiin keskittymään toiminnallisuuteen. Ulkoasun toteutuksesta tuli hieman karu, mutta toimiva ja siisti. Oleellisimmat tiedot tilauksista ovat selkeästi näkyvillä, ja lisätietoja saa painamalla ”AVAA”-tekstiä.

Yksinkertaisuus auttoi myös sovelluksen tekemisessä riippumattomaksi selaimesta ja selaimen asetuksista. Ei tiedetty, miltä tietokoneelta sovellusta tultaisiin käyttämään milloinkin. Siksi oli etu saada sovellus toimivaksi ympäristöstä riippumatta. Työkalu läpäisi yksinkertaiset testit Internet Explorer-, Firefox- ja Chrome-selaimilla. JavaScriptin pois kytkeminen selaimesta teki työkalun esitysasusta pidemmän ja hankalammin luettavan, mutta ei vaikuttanut tietojen hakemiseen muulla tavalla. Tämä katsottiin hyväksyttäväksi, sillä JavaScript on nykyään kytketty pois päältä selaimissa melko harvoin.



Kuva 8. Uuden työkalun oletusnäkymä.

Kuten kuvasta 8 voi huomata, tiedot tilauksista saatiin koottua yhteen, helposti luettavaan paikkaan. ”Laskutettu?”-nappia painamalla voidaan muuttaa tilauksen status käsittelemättömästä käsitellyksi, jolloin tilauksen väri muuttuu vihreäksi. Nappi vaikuttaa suoraan tietokantaan, joten mitään erillisiä toimenpiteitä ei vaadita tietokannan ajan tasalla pitämiseksi. Oleellisimmat tiedot ovat suoraan näkyvillä, mutta tarvittaessa saadaan lisätietoja painamalla ”AVAA”-tekstiä. Käyttäjä voi halutessaan tehdä PDF-laskun suoraan painamalla ”Tee lasku”-nappia.

5 Startup-kulttuurin vaikutus casessa

Startup-hengen mukaisesti ohjelman kehityksessä toteutuksen nopeus osoittautui tärkeämmäksi kuin sen laatu. Ohjelman käyttäjät olivat samoissa tiloissa työskenteleviä myyjiä. Heille pystyttiin tarvittaessa selittämään suoraan, miten ohjelma toimii, vaikka ohjelma ei olisikaan ollut kaikissa kehityksen vaiheissa täydellisen selkeä. Tärkeää oli saada uudet toiminnot käyttöön. Ei myöskään tiedetty, täytyykö koko ohjelma poistaa ja projekti aloittaa uusista lähtökohdista seuraavan viikon aikana. Käyttöön ei näiden syiden vuoksi otettu suunnittelumalleja tai edes luokkia: ohjelmaa alettiin rakentamaan mahdollisimman nopeista lähtökohdista.

Yrityksellä ei käytännössä ollut kuin minimaaliset resurssit tätä projektia varten. Firmalla ei ollut tarjota tietokonetta kehitykseen, vaan työ tehtiin omalla kannettavalla. Kenellekään ei maksettu palkkaa vielä casen aikoihin. Ohjelmointiryhmään kuului vain kaksi henkilöä, ja toisella työntekijöistä oli kädet täynnä itse verkkopalvelun kehityksen kanssa. Kaikki verkkopalvelussa käytetyt teknologiat olivat ilmaisia ja kaikille vapaasti saatavissa. Tämä johtui ilmaisten ratkaisujen riittävyydestä yrityksen tarpeisiin, mutta varmasti myös osaksi siitä, ettei yritys ollut vielä saanut siemenrahaa. Kaikki yrityksen työntekijät tekivät töitä ilmaiseksi täyspäiväisesti. Yrityksen kalleimmat ohjelmistot, Adoben Muse ja Photoshop, olivat yrityksen designerin henkilökohtaisesti omistamia. Maksullisiin lisensseihin ei vain ollut varaa. Ilmaiset teknologiat tosin täyttivät yrityksen tarpeet erinomaisesti.

Työskentelyssä ei käytetty mitään aikaisemmin esitetyistä ketterän kehityksen ohjelmistokehitysmalleista tai mitään muutakaan mallia. Tässä tapauksessa päätöksessä oli järkeä: sovellusta teki täysiaikaisesti vain yksi ihminen. Tällä työntekijällä oli lyhyt sopimus yrityksen kanssa, joten hän ei ollut täysin vakiintunut yritykseen eikä edes tavannut muita it-puolen ihmisiä. Lisäksi projekti oli työmäärältään melko pienikokoinen. Ohjelmistoprojektin viitekehyksen käyttäminen tässä projektissa olisi ollut ylilyönti, paitsi ehkä harjoittelun vuoksi. Voidaan kuitenkin päätellä, että jos kehitystiimi sekä projekti olisivat olleet suurempia ja työntekijät vakiintuneempia, yrityksen olisi ehdottomasti kannattanut ottaa käyttöön Scrum tai TDD. Vaikka casessa ei näitä malleja käytetty, ne oli tärkeää esitellä tässä opinnäytetyössä. Agile-menetelmät toimivat startupeissa erityisen hyvin.

Vaikka nopeus olikin projektin tärkeimpiä asioita, on syytä alleviivata että kommentointia ei pitäisi laiminlyödä, vaikka se hidastaisikin työn etenemistä. Hoputetuissa projekteissa koodin laatu usein kärsii ja nopeat, heikolla pohjalla olevat ratkaisut yleistyvät. Uuden ohjelmoijan on hyvin vaikea päästä mukaan hätäisesti rakennettuun ohjelmistoon ilman selityksiä koodissa. Kommentoinnin tärkeys siis kasvaa, mitä vähemmän aikaa projektiin on annettu. Erityistä huomiota tulisi kiinnittää taikanumeroiden (magic numbers) ja globaalien muuttujien selittämiseen, jos sellaisia on jouduttu käyttämään. Voi tosin olla, että kommentointi jarruttaa projektia tarpeettomasti. Jos esimerkiksi pienellä yrityksellä on vain yksi it-työntekijä, voidaan olettaa, että kommentoinnin tarpeellisuus vähenee. Silti, jopa itse kirjoitetut koodiosuudet unohtuvat ajan myötä.

Ohjelmistotuotannon alalla on muodostunut vakiintuneeksi toimintatavaksi käyttää projekteissa jonkinlaista versionhallintaa. Casessa ei käytetty minkäänlaista versionhallintajärjestelmää, vaan tiedostoista oli pelkät varmuuskopiot. Tämä oli selkeä virhe. Projektin ja yrityksen koosta riippumatta tulisi aina käyttää vähintään kevyttä versionhallintaa. Versionhallintajärjestelmät ovat muuttuneet niin helpoiksi käyttää ja nopeiksi asentaa, että niiden käyttämättä jättämiselle ei ole tekosyitä. Vaikka casessa ei ollut ongelmia tästä johtuen, versionhallinnan poisjättäminen pohjustaa todellista katastrofia tulevaisuutta ajatellen. Versionhallinnan käyttäminen nopeuttaa myös työntekoa yleisesti, kun kaikki vanhat versiot ovat käden ulottuvilla milloin tahansa. Näin päästään tarkastelemaan vanhoja ideoita ja oppimaan virheistä tehokkaammin.

Sovellukselle tehtiin casessa hyväksymistestaus sen viimeisimmän version käyttöönottoon liittyen. Tehtiin laaja testaus usealla selaimella ennalta määritellyn tapauslistauksen perusteelta. Oli hieman riskialtista tehdä testaus kunnolla vasta viimeisessä versiossa ja pelkästään hyväksymistestauksena. Ratkaisu oli kuitenkin toimiva. Testaukseen ei ollut paljon aikaa, kun työkalu haluttiin mahdollisimman nopeasti käyttöön. Työkalu oli käytössä jo puolessavälissä sen kehitystä, joten epäkohdista ja virheistä saatiin välitöntä palautetta. Tapahtui siis eräänlainen käyttäjätestaus jatkuvasti työkalun koko kehitysprosessin ajan. Työkalun toiminnot eivät myöskään olleet erityisen monimutkaisia, joka helpotti testausta ja vähensi sen tarvetta. Tällainen testaustapa osoittautui projektiin riittäväksi.

Casesta ei tehty kattavaa dokumentaatiota. Tämä katsottiin hyväksyttäväksi, sillä koodista tuli hyvin kommentoitua eikä ohjelma sisältänyt mitään erityisen

monimutkaisia funktioita tai muita prosesseja. Koska yritys on pienikokoinen, kaikki työntekijät saatiin koulutettua ohjelman käytössä sen valmistuttua. Sovelluksen hyviin puoliin kuului myös sen yksinkertaisuus ja yksiselitteiseisyys, jotka vähensivät dokumentaation tarvetta.

Ohjelma onnistui täyttämään yrityksen tarpeet hyvin, joten jatkokehitystä ei tarvittu vaan työvoima vapautettiin muihin projekteihin. Jos jatkokehitykseen olisi lähdetty, olisi tavoitteena ollut Ajaxin hyödyntäminen ohjelmassa. Interaktiivisuus tietokannan kanssa pelkän PHP:n avulla jäi hieman kömpelöksi. Työkalu sisälsi myös paljon optimoimatonta koodia eikä ollut kovin helposti muutettavissa. Ideaalitilanteessa uutta, paremmalla pohjalla olevaa ohjelmistoa olisi lähdetty kehittämään tämän työkalun rinnalle. Työ tämän projektin parissa ei kuitenkaan mennyt missään nimessä hukkaan: työkalusta saatiin odotettu hyöty irti todella nopeasti. Ohjelma toimii myös hyvänä prototyyppinä tuleville mahdollisille uusille työkaluille.

6 Yhteenveto

Opinnäytetyön tarkoituksena oli esitellä startup-yritystä sekä siellä tapahtuvaa työntekoa ja verrata näitä esittelyjä tosielämän casen kanssa. Vaikka itse casen verkkopalvelu oli jo olemassa, projektin toteuttajalle annettiin vapaat kädet siihen tuotettavan lisäosan sekä lisätyökalun suunnittelussa ja toteutuksessa. Tämän vuoksi työ tarjosi hyvän yleiskatsauksen käyttöliittymän suunnitteluun, web-teknologioiden implementoimiseen ja varsinkin nopealla aikataululla tapahtuvaan ohjelmistokehitykseen.

Casen kohdeyrityksen henkilökunnan oli vaikea hakea, tarkastella ja muuttaa tietoja yritykselle asetetuista tilauksista. Tämä johtui tietokantaa käyttävien aputyökalujen puutteellisesta suunnittelusta ja hankalasta käytettävyydestä. Ongelma ratkaistiin tekemällä uudistuksia itse verkkosivujen käyttäjäpuoleen sekä luomalla uusi, käytettävämpi ja selkeämpi aputyökalu. Toimenpiteiden jälkeen tilausten tarkastelu nopeutui huomattavasti ja tarkasteluprosessista tuli paljon helpompi. Myös tilauksiin liittyviä tietoja päästiin päivitysten ansiosta muuttamaan. Voidaan siis sanoa, että asetettuihin tavoitteisiin päästiin ja kaikki keskeiset ongelmat ratkaistiin.

Startup-kulttuurin vaikutus casessa oli suuri. Casessa tapahtunut projekti antoi hyvän esimerkin startupissa työskentelystä. Työn laatu saattoi jäädä osittain puutteelliseksi, mutta toivotut tulokset saatiin nopeasti esiin ja kaikkien ulottuville. Tämäntapainen lisäarvon tuottaminen on erittäin tärkeää startupille, sillä se auttaa nuorta yritystä etenemään askel kerrallaan kohti vakiintunutta bisnesmallia.

Lähteet

1. Robehmed, Natalie. What Is A Startup? http://www.forbes.com/sites/natalierobehmed/2013/12/16/what-is-a-startup/. Forbes.

2. Song, M; Podoynitsyna, K & van der Bijl, H & Halman, J.I.M. 2008. Success Factors in New Ventures: A Meta-analysis. Journal of Product Innovation Management, Osa. 25.

3. Kuusela, Sami. Hupparihörhö ja Bisnesmies. 2013. pp. 15-21. Helsinki : Taloustieto Oy.

4. Lennon, Mark. CrunchBase Reveals: The Average Successful Startup Raises $41M, Exits at $242.9M. 2013. http://techcrunch.com/2013/12/14/crunchbase-reveals-the-average-successful-startup-raises-41m-exits-at-242-9m/. CrunchBase.

5. Jalali, Iman. A Reality Check For Anyone Eager To Work At A Startup. 2014. http://www.entrepreneur.com/article/235767. Entrepreneur.

6. Wei, Jiyan. Lessons Unlearned While Working At A Startup. 2014. http://www.forbes.com/sites/groupthink/2014/03/12/lessons-unlearned-while-working-at-a-startup/. Forbes.

7. Schwaber, Ken and Sutherland, Jeff. Scrum Guide. 2013. https://www.scrum.org/Portals/0/Documents/Scrum%20Guides/2013/Scrum-Guide-FI.pdf. Scrum.

8. Slide Hunter. Free Scrum Task Board PowerPoint Template. 2014. http://slidehunter.com/powerpoint-templates/scrum-task-board-powerpoint-template/. Slide Hunter.

9. Bartley, Richard. Integration Hell? 2012. http://richardbartley.blogspot.fi/2012/06/integration-hell.html. Dad Coder.

10. Shore, James. The Art Of Agile Development: Test-Driven Development. 2010. http://www.jamesshore.com/Agile-Book/test_driven_development.html. The Art Of Agile.

11. Ricardo, Luiz. Is TDD Dead? 2014. http://luizricardo.org/en/2014/05/is-tdd-dead/. State of the Art.

12. Aimonetti, Matt. What Technology Should My Startup Use? 2013. http://matt.aimonetti.net/posts/2013/08/27/what-technology-should-my-startup-use/. Matt Aimonetti.

13. W3Schools. HTML Introduction. 2014. http://www.w3schools.com/html/html_intro.asp. W3Schools.

14. Kyrnin, Jennifer. What's New in HTML5. 2014. http://webdesign.about.com/od/html5/a/html_5_whats_new.htm. about.

15. Saarikumpu, Osmo. Johdanto tyyliehdotuksien käyttöön Web-sivuilla. 2013. http://weppipakki.com/css/tekstit/cssintro.htm. Weppipakki.

16. W3Schools. JavaScript Introduction. 2014. http://www.w3schools.com/js/js_intro.asp. W3Schools.

17. —. PHP 5 Introduction. 2014. http://www.w3schools.com/php/php_intro.asp. W3Schools.

18. PostgreSQL. About. 2014. http://www.postgresql.org/about/. PostgreSQL.

19. DB-Engines. DB-Engines Ranking. 2014. http://db-engines.com/en/ranking. DB-Engines.

20. Si. Why should I use version control? 2013. http://stackoverflow.com/questions/1408450/why-should-i-use-version-control. Stack Overflow.